Parsian
(Amirkabir Univ. of Technology Robocup Small Size Team)
Team Description for Robocup 2007

Amir Amjad Amir Ghiasvand, Mohammad Korjani, Mani Milani, Vali Allah Monajjemi, Mohsen Roohani, Hamed Gha'ed Nia

Electrical Engineering Department, Amirkabir University of Technology (Tehran Polytechnic)
424 Hafez Ave. Postal Code 158754413, Tehran, Iran
webmaster@parsianrobotic.com

Abstract. This document describes Parsian Robotic Small Size RoboCup team activities for preparing for Robocup 2007 Small Size League. It's an overview of all activities done in past one and a half year. It's the second time that Parsian is participating in SSL. Our robots mechanical and electrical design, wireless communication, image processing, AI algorithms and software are described in this paper.

1 Introduction

Parsian Robotic is a robotic research group working in Electrical Engineering department of Amirkabir University of technology (Tehran Polytechnic). After being qualified for RoboCup 2006 Germany, This is the second year of Parsian's activity in small size robots' field.

2 Hardware Architecture

2.1 Mechanical Design

Our current robot is a three wheel omni directional robot with angle of 120 deg between wheels, The diameter is 60.5mm and the robot consist of two floor of aluminum plate and fastened together by use of three columns with a hole inside where the motor are placed. The shooter consists of a regular solenoid that is attached at the top plate and the motion is transmitted by a metal bar to the shooter head. The roller is a cylinder mounted on two bearings and driven by a dc motor as shown in the picture.

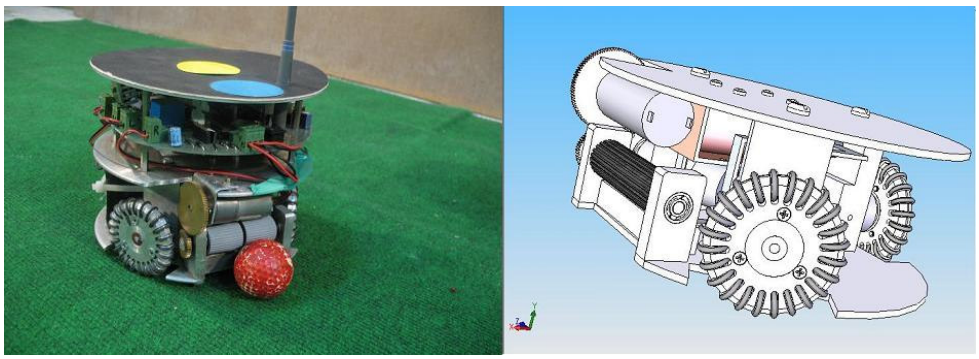


Fig. 1. Our 3rd generation robot (left), Mechanical Overview (right)

2.1.1 Wheels

Our new omni wheel is shown in the figure 2. This kind of wheel became very popular in recent years. These wheels improve robots' friction with field and smooth the omni movement of the robot.

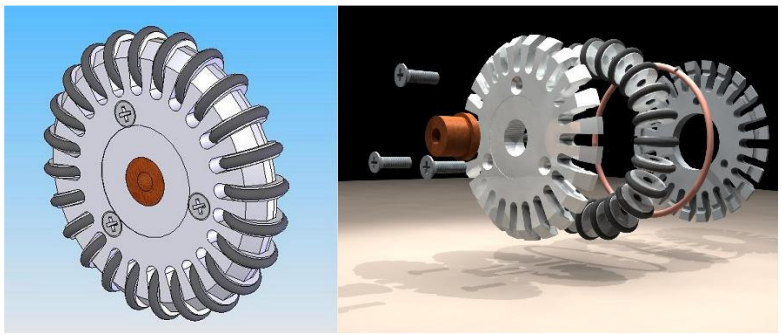


Fig. 2. Our new omni directional wheels

2.1.2 Robots Geometrical Properties

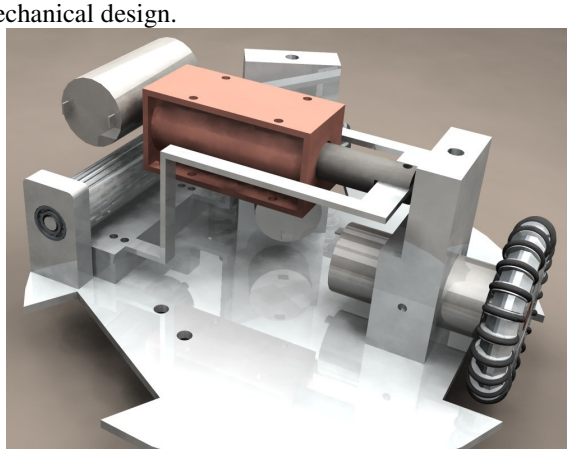
Diameter: *178 mm*
Height: *143.8 mm*
Ball coverage (due to RoboCup rules): *19%*
Weight: *1.8~1.9 Kg*

2.1.3 DC Motors

We are using "*25GA-RETARDING GEAR MOTOR"* which are 12 volt DC motors with gear box (1:11.8) .The motor speed is 500rpm that leads to linear speed of 1.5 m/s. We will use "*Faulhaber dc motor 2233V024S"* in the next generation of robots to increase speed and accuracy. This will be discussed further in this document.

2.1.4 Next Generation

We are planning some major improvements which will be implemented in the next generation of mechanical design.



Weight Reduction: In order to improve motor speed and lower inertia forces we are working hard on new materials to reduce total robot's weight. Weight reduction also makes angular movement more accurate. Removing extra parts would provide more space for better design.

New DC motors: In order to reach higher speed and reducing energy consumption in our robots we decided to change our dc motors. We will use "*Faulhaber Motors 2224U006SR*" for wheel drive and *2342S006CR* for spinner like many other teams; these motors are reasonable choices because of high power and small size. The main motor is a 9000rpm dc motor which is needed to be used with spur gears to reduce the speed. By using different gears for keeper, defenses and forwards we can achieve speeds needed for every position; for example keeper needs higher speed comparing to forwards.

New pass and shoot method: This year mechanical group decided to develop a new method of shooting and passing that has not been ever used (as far as we know). The main idea is to separate the angle of the robot and the angle of shooting system – in limited range of variation -. We call this method "*rotational shooting*".

The idea is to make a box consisting of the shooter and the spinner and then separating the shooter from this box by giving it one additional rotational degree of freedom. This rotational joint can be placed in two different positions and thus can be built by two different strategies, first to place the joint in the front of the robot and second to place the joint in the center of it.

By placing the joint in the center of robot the space occupied due to motion of shooter is at the front where it's easer to find that space but building this joint in the center of robot is much more difficult (*figure 3-left*), But if the joint is placed in the front, the space for the motion is harder to find but the joint is easier to build (*figure 3-right*). This motion can be taken out from a four bar mechanism which the shooter freezes the coupler bar, thus can lead to appropriate motion (*figure 4*)

Another way is that the shooter can be driven by a worm gear so it is auto locked as desired.

The advantage of using this method of shooting is that all teams predict the direction of shooting by drawing a line using center of robot and ball and the keeper and defenses location are adjusted according to that and this method can mislead the opponent robots especially in penalty and free kicks.

2.1.5 Chip Kick

As many teams are using chip kicks in small size competition we decided to develop that system for some are our robots as discussed before we should choose to use One method between rotational shooting and chip kick shooting because they can not come together in one robot so we decided to use the chip kicking for our defenses and rotational shooting for forwards.

Our chip kick mechanism is to lift the ball up just before the shoot; it is a very common way for chip kicks. The final decision about chip kick will be made after making and testing the rotational shoot.

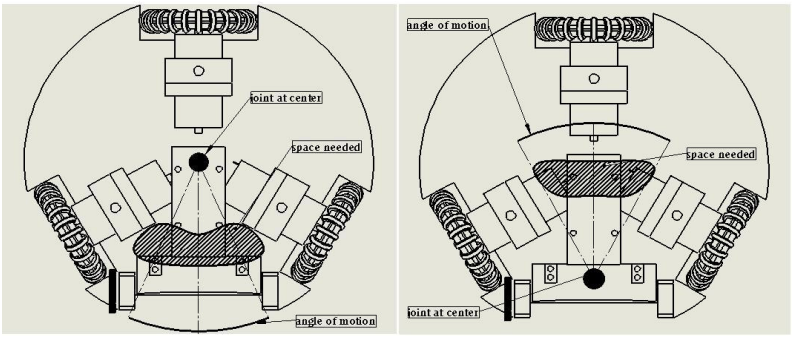


Fig. 3. New Method for passing

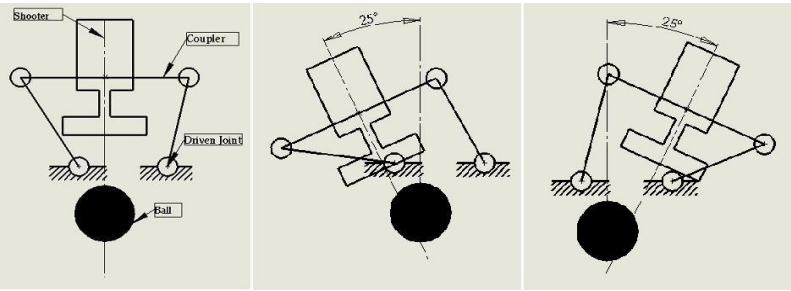


Fig. 4. 4-bar mechanism for moving the shooter

2.2 Electrical Design

Electronic circuit consists of three parts:

- Communication receiver
- Motor and Spinner driver
- Kicking System Driver

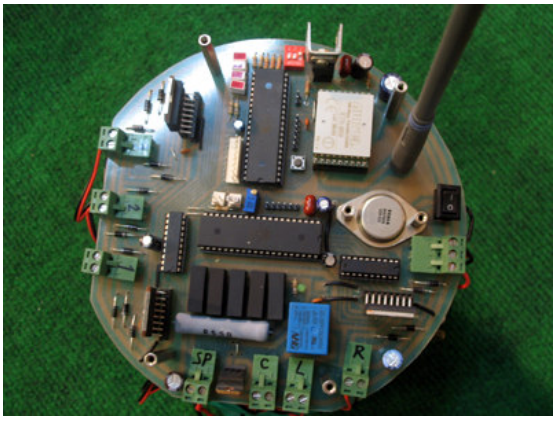


Fig. 5. Main Circuit

As demonstrates in (*figure 5)* circuit board is industrial standards' compliant. For increasing the reliability of the system, each part of the circuit have separate power supplier.

Communication part includes AUREL XTR module and attached AVR microcontroller. This circuit operates in 434/869 MHz frequency and manages data reception. After receiving data from transmitter, the microcontroller delivers decoded commands to the motor driver section through serial protocol.

In the driver module, three motors are driven using two LM298s. The microcontroller first processes the input signal of shaft encoders and then controls the speed of each motor through generating proper *Pulse Width Modulated* signals.

A simple infrared sensor in front of the robot senses the ball presence and triggers the microcontroller to start the spinner.

In the kicking part, we use 5 DC to DC converters to increase the voltage from 18 to 180 volts. This potential charges two 2200 micro farads capacitors. When kick signal receives the microcontroller drives a relay which opens the discharge path to the solenoid. The microcontroller adjusts the relay opening time in order to control the kick strength based on the received command.

2.3 Wireless Communication

The wireless communication system consists of 3 main parts:

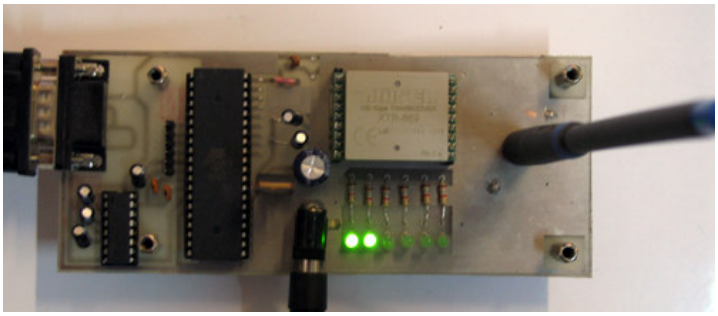


Fig. 6. Transmitter Module

2.3.1 Transceiver module

We use "*AUREL XTR-434*" and "*AUREL XTR-869*" transceiver modules for data transmission and reception. The XTR-434 uses the carrier frequency of 433.92 MHz and XTR-869 uses 869 MHz carrier frequency. Both work with bandwidth of 150 KHz. Both can work in data frequencies between 9600 - 100,000 bps, but it is said not to be used for frequencies less than 57600 when not applying any bit-balancing algorithms to the data, though the data must be byte-balanced otherwise.As mentioned the only difference between these modules is the carrier frequency. As it's vital for Robocup robots to be able to work in two different carrier frequencies, two different modules are used, they can be easily get replaced to change the team carrier frequency during matches.

The XTR-434/869 transceiver can be set to transmitter, receiver and power-down modes. When switching to transmitter or receiver, the device needs 2ms to switch to the new mode. And for 1ms after it should not be used for data transmission, but a rectangular pulse signal must be transmitted in the meantime.

As we did not use the available manufactured PCBs of the XTR-434/869 module, we came across some problems using our self-made circuit board. The correct way of placing the module on a circuit board, the antenna connection, the ground circuits and the antenna which is used, as said in the user's manual, is noticeable for better performance!

2.3.2 Data Flow Control

Transmitter: there is an AVR microcontroller in the transmitter circuit of the wireless communication board which is used to read the original data from the Control unit computer and to create a packet of processed data for the XTR-434/869 to transmit.

The serial Rx port of the AVR microcontroller has an interrupt subroutine which is called anytime it receives one byte of data. When we receive 30 bytes of data, 5 bytes for each robot, we apply the bit-balancing algorithm to the 30 original-data-bytes giving out 60 bytes of processed data with equal number of 1s and 0s in each byte.

The "rf_transmit" function is then called to create a packet containing 1ms of rectangular pulse signal plus one start-byte plus two bytes containing the number of the bytes to be transmitted. Then the bit-balanced data followed by two ending bytes are added to the created packet. The whole packet is sent to the XTR-434/869 TX port, using the serial TX port of the AVR microcontroller for transmission.

Receiver: there is an AVR microcontroller on the receiver circuit board of the wireless communication system on each robot which gets the received packet from the XTR-434/869 RX port. It then decodes the received packet in order to reconstruct the original data. The data is checked for any errors using 3 different procedures, before being passed to the next unit.

2.3.3 Serial Connection to PC

The serial RS232 COM port of the Control unit computer is used to send data to and receive any needed data from the AVR microcontroller. A MAX-232 IC is used to convert the +12 and -12 volts of the COM port to the operating logic voltages of the circuit elements, that is 0 and 5.

3 Software

3.1 Main Software

The main software application is written in Visual C++. This application performs three main tasks, Robots control, Strategy Layer Implementation (Game Play) and Simulation.

Here is the software general schema:

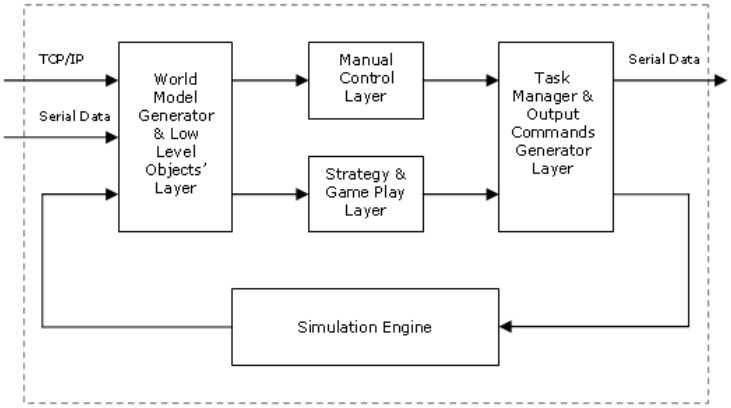


Fig. 7. Software Schema

3.1.1 World Model Generator & Low Level Objects' Layer

The main task of this layer is receiving data (Image Processing & Referee Box) from real world and defining an accurate and reliable world model based on those data.

In each program cycle, received data first gets verified. Then this layer generates the world model. This world model consists of robots' and ball positions. This layer also applies proper transformations to the received data to neutralize direction changes, e.g. after half time. This layer then sets/unsets several flags for each robot and the game state based on received data. Other layers have no access to any other information except what supplied by this layer.

When running software in simulation mode, there is no real connection between the software and the real world. Instead, the simulation engine simulates these relations.

3.1.2 Manual Control Layer

In development phase, it's essential to have several tools to test every aspect of robots, e.g. movement, passing, shooting and co-operating. These tools are available in this mode. The software bypasses strategy layer in this mode and all robots in the field (real or simulated) can be fully controlled by the operator.

3.1.3 Strategy & Game Play Layer

This layer is the most important layer for the act of playing soccer. Briefly, this layer decides "who do what" based on the world model and game state. It will be fully described further in this document. (See section 3.2)

3.1.4 Task Manager & Output Commands Generator Layer

Outputs of both manual control and strategy layer are “Commands”; these commands first stay in a queue. We named this queue “Action Queue”. This layer fully manages this queue for each robot. Several algorithms used to reduce the conflicts and increase the efficiency of every robot. As every action in queue can have different priority, some other algorithms are implemented to choose the best action. After deciding the optimum action for each robot, this layer finally generates the output commands package and sends the package through serial port to Wireless Communication module.

3.1.5 Simulation Engine

For making the development easier, the simulation engine implemented inside to software to fully simulate a “Small Size Robot Game”. This layer receives commands from the Task Manager Layer and changes the world model data in order to simulate the changes. The simulation engine uses a graphical interface to sketch the virtual world. Feature plans consist of implementing “Open Dynamics Library http://www.ode.org” inside the simulation engine to improve its accuracy and flexibility. Current simulation engine is capable to simulate all the team strategy with more than 70% of reliability factor.

As mentioned before, this application is written in Microsoft Visual C++. We run this application under Microsoft Windows XP on an Intel Pentium IV PC @ 3.06 GHz and with one gigabyte of RAM.

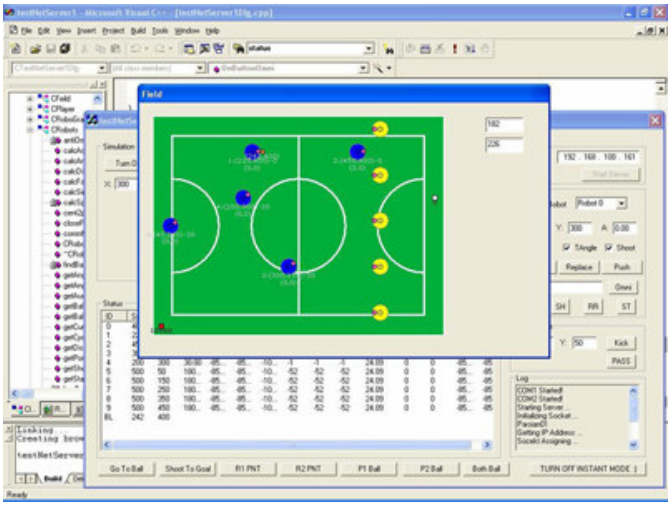


Fig. 8. Main Software Screenshot (Simulation Mode)

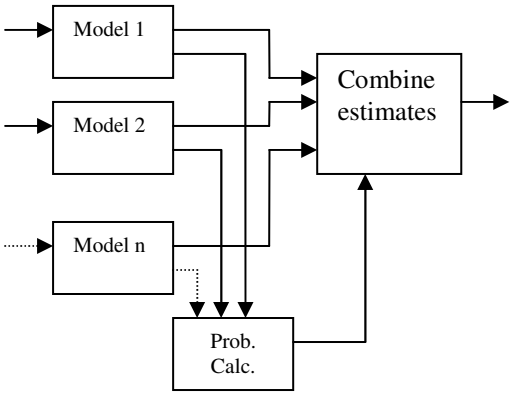
3.2 Game Strategy

The team strategy is based on current world model which is provided by the low level layer of the main software. One of the most important things in strategy implementation is the prediction. Prediction reduces the effect of the system immanent delay when tracking and controlling fast moving robots. Because of physical feedback loop delay in the controlling fast moving robots, we use predictor to cope with the system latency. The pose predictor (orientation and position) of the ball and robots can significantly improve the accuracy and precise of the strategy. In the strategy layer we need not only the current positions but also future game state. In order to predict the future, we apply gradient method based on fuzzy logic to process the position with the last ten frames. Robots behavior changes based on the last position and action command fed to robots and future position of robots and ball.

The strategy has global and local behavior. The global behavior consists of defense, attack and stop mode. Local behavior includes passing, shooting, dribbling, robot placement and tracing the ball. In each global behavior there are several models. The strategy chooses the best model in each situation against opponents. We use 'Multiple Hypothesis Testing' method in order to overcome the problem. For each model, the hypothesis that it is the true model is raised, and the probability that the hypothesis is correct is evaluated using the observation sequence. Over time the most accurate model is assigned the largest probability and so dominates the behavior of estimator. The output is a function of estimates made by the different model as well as the probability that each model is correct. Following two assumptions are made:

- The true model is one of those proposed
- The same model has been in action since t = 0.

Over time, one model acquires a significantly greater probability than the other model, and it corresponds to the model whose parameter values are most appropriate.



By using 'Discrete Fuzzy Role Bases' after getting every robot and the ball state and state differences we defined some 'membership functions' and 'fuzzy role bases'. The output of these fuzzy sets is robots' behaviors, which can be defined by assuming

the current robot states, ball position and opponents' movement method. Here are some sample Fuzzy role bases which can define the behaviors (global or local):

- Global behavior: If the opponent is in attacking state and the ball is in its hand then our robots should be in defense state.
- Local behavior: If the ball is in our robots hand and the closest opponent is far enough then the robot should dribble to the opponent goal.
- Local behavior: If the ball is in our robots hand and the opponent robot is not far enough then our robot should pass the ball to the closest teammate.
- And so on , …

By using these role bases and defining appropriate membership functions we are able to define the best robot behavior model at the time. As these models have been defined as Fuzzy Behaviors, The final approach is the best behavior for the robot.

This method is one the best methods comparing to other AI and Intelligent Search Methods, with higher calculation and expanding ability. By using this Multi Agent System we can teach the system the new behaviors we may experience.

Positions' variables and their differences x,y,x',y' are used to calculate the angle and the velocity of the moving objects(Robots and the ball). By using difference of states (positions) this system has the 'Prediction' ability. So it can predict the ball and opponents' robots positions and redefine every robot behaviors based on those parameters.

After defining every robot behavior, by using another Fuzzy system we can convert every robot behavior into the target position, it means we can define every robot target by using the calculated behaviors as an input for another fuzzy system.

Inputs of the 'Routing System' are current robots' positions and every robot target position. This section would produce every robot path by using 'Half Field Neural Network' in a fuzzy environment. Optimizing the path is vital for reducing the robots' energy. For optimizing the path using half field neural network we use 'energy minimizing method' based on '*Liapunov stability theorem*'.

Finally, this layer fills an array with proper decisions for each robot. These data then being used by Task Manager Layer.

4 Vision System

Studying some different documents and data -sheets and parameters of different cameras, we decided to use a CCD-Analogue camera. Some important parameters of the camera to use are discussed below.

The resolution of the camera output video must be at least 640x480 pixels so that the image-processing algorithm can give a better and more delicate output. The resolution of the grabbed frames sets a minimum limit to the capture device specifications which is discussed later. The more the shutter speed of the camera is, the more blur-less will be the image that it provides. This helps2 the noise-reduction algorithm, to give a more definite understanding of the playing field. It was important

to pay attention to the frame-rate of the output video of the camera, though most of the studied cameras would meet our need.

Our vision system is now based on two "*Panasonic 3 Color CCD WV CT460/G*" Cameras. These RGB cameras operate with speed of 30 frames per second.

The cameras' outputs then get captured by use of a standard frame grabber. "*DirectX9.0 SDK DirectShow Libraries*" then used as the main framework for the image processing software. This software is also written in VC++.

In order to identify each robot distinctively, we used some different patterns of circles that we painted in one of those 12 colors. Each robot is distinctively identified by these colored circles that are placed on top of each robot, considering the arrangement of the circles.

In order to locate the circles of each color we thought of some different ways of processing the matrix of the image among which I will briefly explain two most inspiring ones.

- One way of locating the circles is to read all the pixels of the grabbed frame without bypassing any important or redundant pixels. In this case, a chosen pattern that best represents a circle will scan the whole image and find all the circles and finally drop the invalid circles found. Note that in this procedure quite a lot of CPU usage and memory space will be used. This will cause the program not to be able to produce as many new frame data as we need.
- Another way of locating the circles is to read not ALL the pixels of the image matrix. In this case a different pattern will sweep the whole picture, but dropping about 95% of the pixels, if they don't contain important material. This is the procedure we are using now.

To identify each robot and find the angles needed, the algorithm goes through some routine mathematical calculations then.

Each camera connects to a separate PC. Each PC runs the same software to process the image and sends its data through TCP/IP to the merging server program. This program merges collected data into one pre-world model. The merged data finally sends the information through TCP/IP to the software PC.

It's obvious from this document that at least three PCs are needed beside our team of robots, two for image processing and one for game play. These three PCs must be connected togethet through high speed TCP/IP network.

References:

1. Siegwart , Roland : Introduction to Autonomous Mobile Robots , 2004, MIT Press
2. Nils J. Nilsson : Introduction to Machine Learning, Department of Computer Science, Stanford University